Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра ІПІ

Звіт

з лабораторної роботи № 1 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

„ **Проектування і аналіз алгоритмів внутрішнього сортування**”

Виконав(ла) *ІП-13 Шевцова Анастасія Андріївна*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов Олексій Олександрович*
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

2 ЗАВДАННЯ

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям (таблиця 2.1):

– стійкість;

– «природність» поведінки (Adaptability);

– базуються на порівняннях;

– необхідність додаткової пам'яті (об'єму);

– необхідність в знаннях про структуру даних.

Записати алгоритм внутрішнього сортування за допомогою псевдокоду (чи іншого способу по вибору).

Провести аналіз часової складності в гіршому, кращому і середньому випадках та записати часову складність в асимптотичних оцінках.

Виконати програмну реалізацію алгоритму на будь-якій мові програмування з фіксацією часових характеристик оцінювання (кількість порівнянь, кількість перестановок, глибина рекурсивного поглиблення та інше в залежності від алгоритму).

Провести ряд випробувань алгоритму на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів) і різних наборів вхідних даних (впорядкований масив, зворотно упорядкований масив, масив випадкових чисел) і побудувати графіки залежності часових характеристик оцінювання від розмірності масиву, нанести на графік асимптотичну оцінку гіршого і кращого випадків для порівняння.

Зробити порівняльний аналіз двох алгоритмів.

Зробити узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	**Алгоритм сортування**
1	Сортування бульбашкою
2	Сортування гребінцем («розчіскою»)

3 ВИКОНАННЯ

3.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму сортування бульбашкою та гребінцем на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

Властивість	**Сортування бульбашкою**	**Сортування гребінцем**
Стійкість	так	так
«Природність» поведінки (Adaptability)	ні	ні
Базуються на порівняннях	так	так
Необхідність в додатковій пам'яті (об'єм)	ні	ні
Необхідність в знаннях про структури даних	так	так

3.2 Псевдокод алгоритму

Сортування бульбашкою:

підпрограма bubbleSort(array, size)
    повторити для i **від** 0 **до** size - 1, **збільшувати на** 1
        повторити для j **від** 0 **до** size - i - 1, **збільшувати на** 1
            якщо array[j] > array[j + 1] **то**
                tmp := array[j]
                array[j] := array[j + 1]
                array[j + 1] := tmp
            все якщо
        все повторити
    все повторити
все підпрограма

Сортування гребінцем:

```
підпрограма combSort(array, size)
    factor := 1.247
    gapFactor := size / factor
    якщо gapFactor > 1 то повторити
        gap := int(gapFactor)
        повторити для i від 0 до size - gap, збільшувати на gap
            якщо array[i] > array[i + gap] то
                tmp := array[i]
                array[i] := array[i + gap]
                array[i + gap] := tmp
            все якщо
        все повторити
        gapFactor /=factor
    все якщо
все підпрограма
```

3.3 Аналіз часової складності

Сортування бульбашкою:

Найгірший випадок $O(n^2)$

Найкращий випадок $\Omega(n^2)$

Середній випадок $\Theta(n^2)$

Сортування гребінцем:

Найгірший випадок $O(n^2)$

Найкращий випадок $\Omega(n)$

Середній випадок $\Theta(nlogn)$

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

```python
from random import randint
from sort_algorithms import *

size = int(input('Введіть розмір масиву: '))
arr_best = [i for i in range(size)]
arr_worst = [size-1-i for i in range(size)]
arr_rand = [randint(0,size) for i in range(size)]

print('\nНайкращий випадок\n', arr_best)
n_comp_b, n_perm_b = alg_choice(arr_best)
print('Кількість порівнянь - ', str(n_comp_b))
print('Кількість перестановок - ', str(n_perm_b))
print('Відсортований масив\n', arr_best)

print('\nНайгірший випадок\n', arr_worst)
n_comp_w, n_perm_w = alg_choice(arr_worst)
print('Кількість порівнянь - ', str(n_comp_w))
print('Кількість перестановок - ', str(n_perm_w))
print('Відсортований масив\n', arr_worst)

print('\nВипадкове заповнення\n', arr_rand)
n_comp_r, n_perm_r = alg_choice(arr_rand)
print('Кількість порівнянь - ', str(n_comp_r))
print('Кількість перестановок - ', str(n_perm_r))
print('Відсортований масив\n', arr_rand)
```

```python
def bubble_sort(arr):
    n_comp = 0
    n_perm = 0
    size = len(arr)
    for i in range(size - 1):
        for j in range(size - i - 1):
            n_comp += 1
            if arr[j] > arr[j + 1]:
                tmp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = tmp
                n_perm += 1
    return n_comp, n_perm

def comb_sort(arr):
    n_comp = 0
    n_perm = 0
    size = len(arr)
    factor = 1.247
    gap_factor = size / factor
    while gap_factor > 1:
        gap = int(gap_factor)
        for i in range(0, size - gap, gap):
            n_comp += 1
            if arr[i] > arr[i + gap]:
                tmp = arr[i]
                arr[i] = arr[i + gap]
                arr[i + gap] = tmp
                n_perm += 1
        gap_factor /= factor
    return n_comp, n_perm
```

```python
def alg_choice(arr):
    alg = input('Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): ')
    n_comp, n_perm = 0, 0
    if alg == 'b':
        n_comp, n_perm = bubble_sort(arr)
    elif alg == 'c':
        n_comp, n_perm = comb_sort(arr)
    else:
        print('Неправильна буква!')
    return n_comp, n_perm
```

3.4.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

Рисунок 3.1 – Сортування масиву на 100 елементів

Сортування бульбашкою:

```
Введіть розмір масиву: 100

Найкращий випадок
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  4950
Кількість перестановок -  0
Відсортований масив
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99]

Найгірший випадок
 [99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70
, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40
, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10
, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  4950
Кількість перестановок -  4950
Відсортований масив
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99]

Випадкове заповнення
 [58, 46, 7, 41, 88, 53, 44, 0, 73, 11, 72, 33, 85, 28, 92, 71, 26, 40, 30, 74, 41, 20, 22, 32, 20, 69, 95, 77, 65, 38,
70, 6, 65, 92, 92, 16, 83, 82, 11, 6, 90, 62, 76, 31, 48, 24, 32, 89, 26, 76, 70, 78, 84, 6, 91, 78, 66, 43, 82, 71, 43,
 78, 78, 27, 19, 16, 58, 41, 72, 65, 77, 65, 74, 50, 17, 80, 42, 15, 100, 28, 23, 52, 80, 29, 89, 60, 44, 76, 74, 21, 2,
 1, 31, 98, 48, 14, 82, 63, 55, 38]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  4950
Кількість перестановок -  2436
Відсортований масив
 [0, 1, 2, 6, 6, 6, 7, 11, 11, 14, 15, 16, 16, 17, 19, 20, 20, 21, 22, 23, 24, 26, 26, 27, 28, 28, 29, 30, 31, 31, 32, 3
2, 33, 38, 38, 40, 41, 41, 41, 42, 43, 43, 44, 44, 46, 48, 48, 50, 52, 53, 55, 58, 58, 60, 62, 63, 65, 65, 65, 65, 66, 6
9, 70, 70, 71, 71, 72, 72, 73, 74, 74, 74, 76, 76, 76, 77, 77, 78, 78, 78, 78, 80, 80, 82, 82, 82, 83, 84, 85, 88, 89, 8
9, 90, 91, 92, 92, 92, 95, 98, 100]
Press any key to continue . . .
```

Сортування гребінцем:

```
Введіть розмір масиву: 100

Найкращий випадок
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  533
Кількість перестановок -  0
Відсортований масив
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99]

Найгірший випадок
 [99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70
, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40
, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10
, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  533
Кількість перестановок -  493
Відсортований масив
 [19, 83, 79, 84, 85, 69, 75, 57, 87, 74, 47, 71, 81, 67, 65, 77, 80, 72, 64, 60, 76, 49, 59, 33, 36, 63, 70, 48, 68, 21
, 54, 61, 51, 55, 62, 39, 50, 43, 17, 53, 56, 31, 34, 44, 52, 15, 35, 42, 45, 73, 46, 41, 29, 88, 27, 37, 40, 14, 38, 30
, 0, 9, 22, 24, 32, 11, 23, 25, 28, 82, 26, 18, 8, 86, 78, 58, 20, 12, 3, 13, 16, 7, 89, 6, 4, 90, 10, 5, 1, 66, 93, 91,
 94, 92, 2, 95, 96, 97, 98, 99]

Випадкове заповнення
 [60, 27, 21, 68, 60, 92, 43, 21, 35, 11, 66, 1, 39, 82, 17, 96, 49, 45, 76, 65, 57, 40, 63, 0, 48, 33, 16, 54, 82, 66,
86, 65, 14, 25, 93, 31, 99, 32, 61, 91, 47, 2, 61, 39, 79, 12, 0, 93, 2, 99, 46, 73, 83, 18, 24, 73, 79, 9, 82, 52, 41,
43, 25, 3, 80, 39, 34, 60, 54, 47, 35, 68, 86, 49, 95, 3, 78, 19, 33, 1, 33, 97, 12, 55, 38, 5, 36, 73, 48, 89, 70, 80,
37, 75, 36, 83, 26, 37, 10, 50]
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  533
Кількість перестановок -  420
Відсортований масив
 [2, 11, 21, 27, 17, 35, 39, 21, 1, 33, 16, 40, 43, 14, 45, 48, 57, 54, 31, 63, 0, 46, 47, 25, 3, 60, 65, 66, 65, 33, 12
, 61, 2, 66, 32, 41, 68, 49, 12, 0, 39, 54, 39, 73, 73, 38, 76, 9, 24, 79, 18, 79, 35, 60, 80, 25, 52, 5, 43, 47, 34, 70
, 19, 3, 60, 48, 78, 82, 68, 82, 49, 82, 80, 83, 61, 33, 1, 73, 26, 36, 55, 37, 86, 75, 83, 86, 89, 36, 10, 50, 91, 92,
93, 93, 37, 95, 96, 97, 99, 99]
Press any key to continue . . .
```

Рисунок 3.2 – Сортування масиву на 1000 елементів

Сортування бульбашкою:

```
Введіть розмір масиву: 1000

Найкращий випадок
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  499500
Кількість перестановок -  0

Найгірший випадок
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  499500
Кількість перестановок -  499500

Випадкове заповнення
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): b
Кількість порівнянь -  499500
Кількість перестановок -  249935
Press any key to continue . . .
```

Сортування гребінцем:

```
Введіть розмір масиву: 1000

Найкращий випадок
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  5603
Кількість перестановок -  0

Найгірший випадок
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  5603
Кількість перестановок -  5541

Випадкове заповнення
Натисніть літеру, щоб вибрати алгоритм сортування. (бульбашкою - b, гребінцем - c): c
Кількість порівнянь -  5603
Кількість перестановок -  5386
Press any key to continue . . .
```

3.5 Тестування алгоритму

3.5.1 Часові характеристики оцінювання

В таблиці 3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування бульбашки для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	45	0
100	4950	0
1000	499500	0
5000	12497500	0
10000	49995000	0
20000	199990000	0
50000	1249975000	0

Сортування гребінцем.

Розмірність масиву	Число порівнянь	Число перестановок
10	43	0
100	533	0
1000	5603	0
5000	27842	0
10000	55589	0
20000	110310	0
50000	316100	0

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	45	45
100	4950	4950
1000	499500	499500
5000	12497500	12497500
10000	49995000	49995000
20000	199990000	199990000
50000	1249975000	1249975000

Сортування гребінцем.

Розмірність масиву	Число порівнянь	Число перестановок
10	43	19
100	533	493
1000	5603	5541
5000	27842	27781
10000	55589	55522
20000	110310	110241
50000	316100	316017

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	45	23
100	4950	2329
1000	499500	255972
5000	12497500	6211976
10000	49995000	25048884
20000	199990000	100648884
50000	1249975000	625787780

Сортування гребінцем.

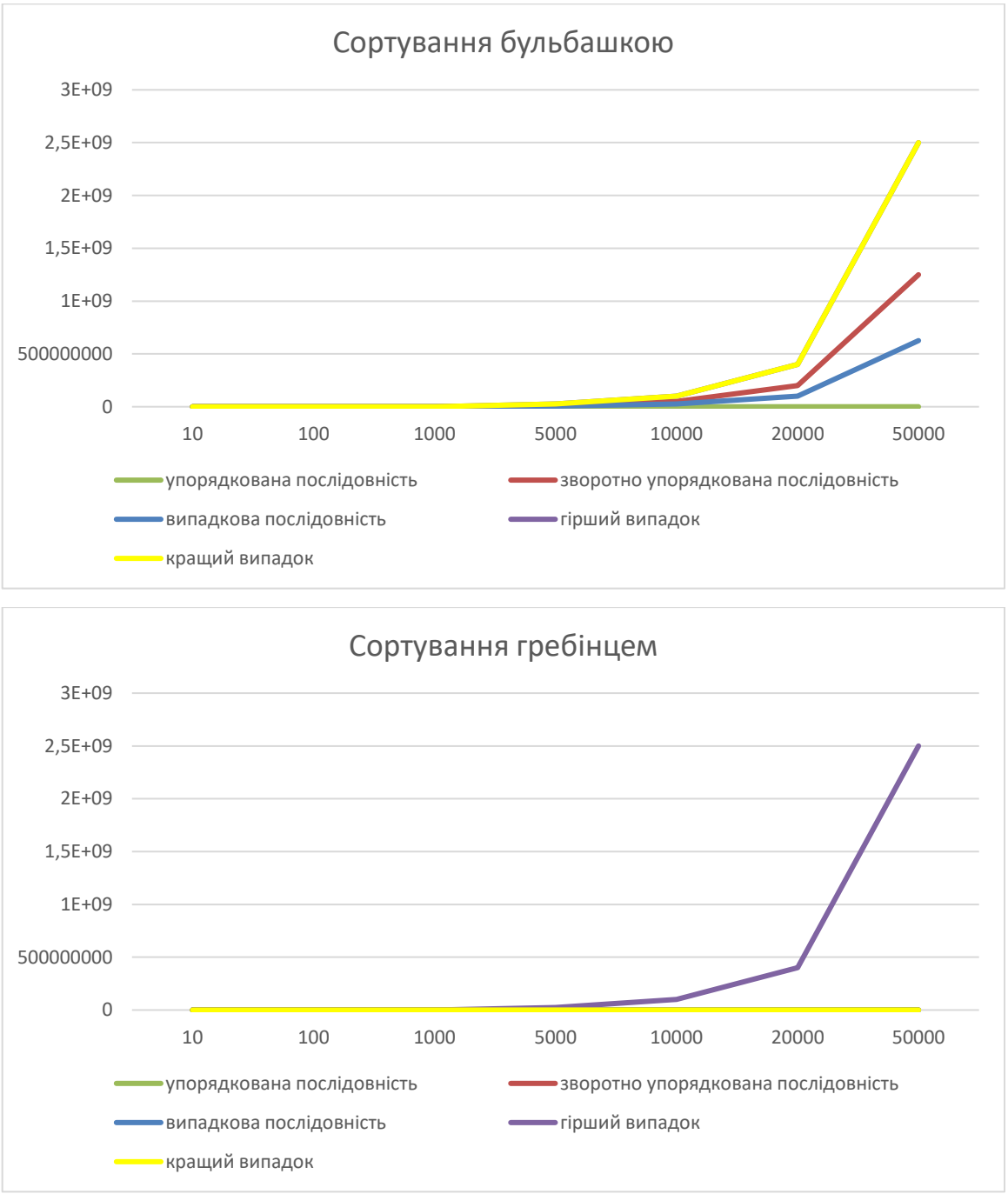
Розмірність масиву	Число порівнянь	Число перестановок
10	43	14
100	533	443
1000	5603	5394
5000	27842	27549
10000	55589	55312
20000	110310	109929
50000	316100	315576

3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять

випадкову послідовність елементів (синій графік), також показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння.

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання



ВИСНОВОК

При виконанні даної лабораторної роботи я ознайомилася з алгоритмами сортування бульбашкою та гребінцем. Дослідила їхню ефективність на масивах різної розмірності.